

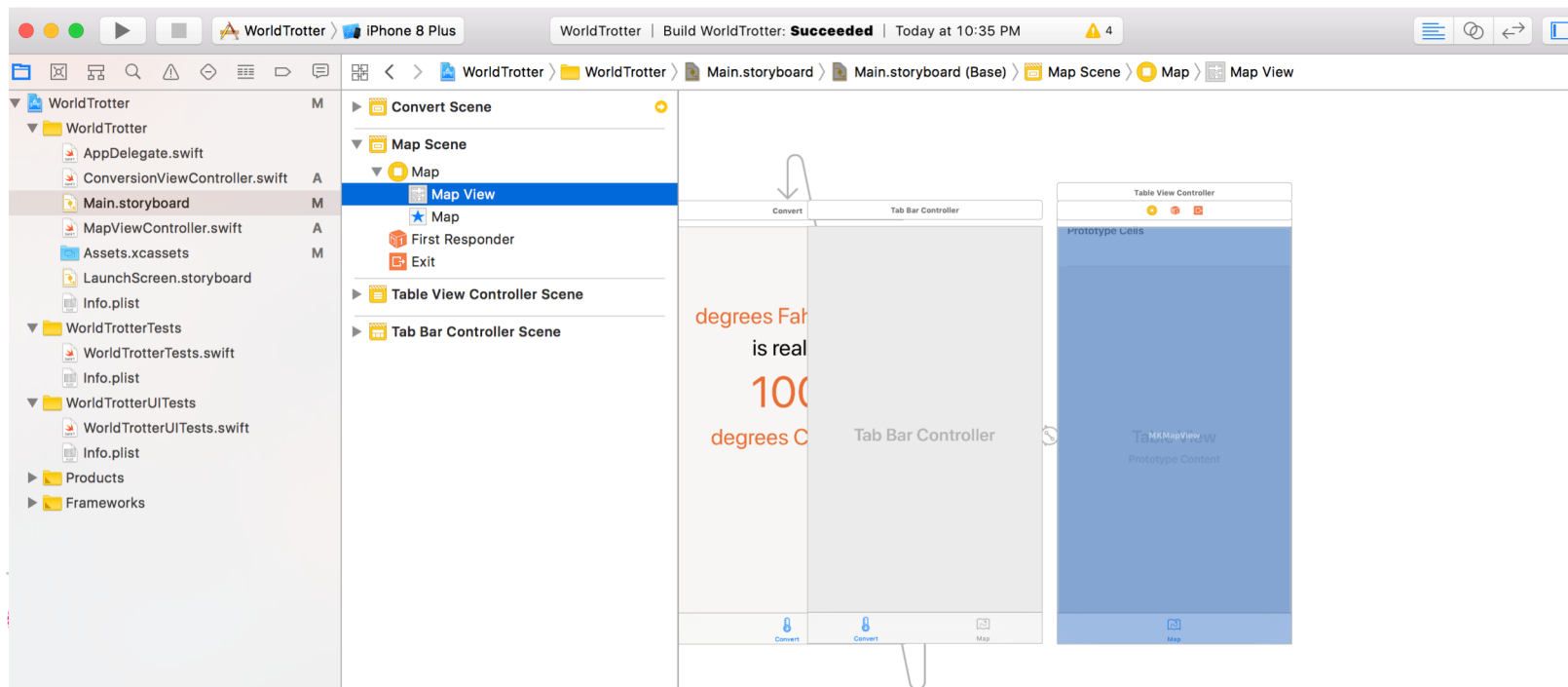
---

# Programming the view for MapViewController



## MapViewController view is currently defined in storyboard

Select map view under Map View Controller and delete it



Open MapViewController.swift

Override loadView() to create instance of MKMapView() to set it

import UIKit

**import MapKit**

class MapViewController: UIViewController {

**var mapView: MKMapView!**

**override func loadView() {**

**//create a map view**

**mapView = MKMapView ()**

**//set it as \*the\* view of this view controller**

**view=mapView**

**}**

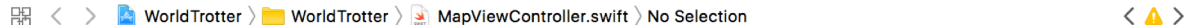
    override func viewDidLoad() {

        super.viewDidLoad()

        print("MapViewController loaded its view")

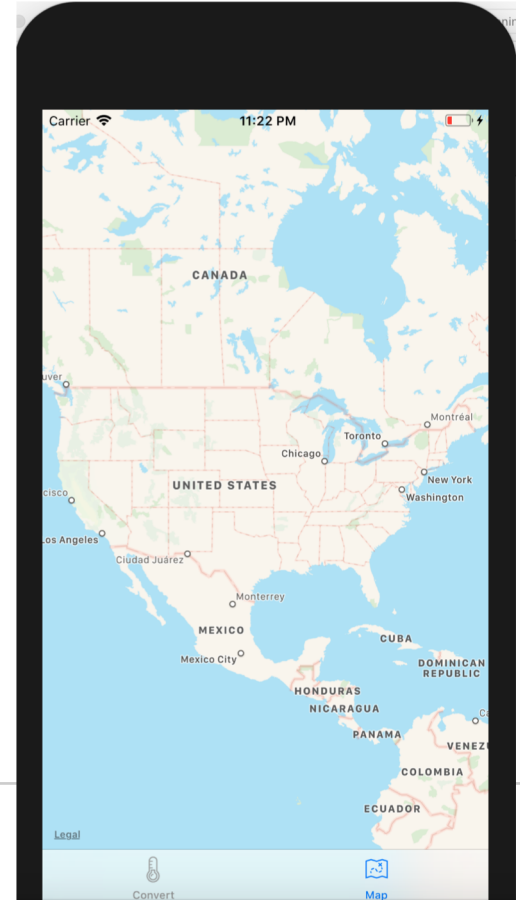
    }

}



```
4 //
5 // Created by daria tsoupikova on 2/3/19.
6 // Copyright © 2019 daria tsoupikova. All rights reserved.
7 //
8
9 import UIKit
10 import MapKit
11 class MapViewController: UIViewController {
12
13     var mapView: MKMapView!
14     override func loadView() {
15         //create a map view
16         mapView = MKMapView ()
17         //set it as *the* view of this view controller
18         view=mapView
19     }
20
21     override func viewDidLoad() {
22         super.viewDidLoad()
23
24         print("MapViewController loaded its view")
25
26     }
27 }
```

Build and run to test. The map is created programmatically at runtime.



If your views are created in Swift, by programming, you need to constrain them programmatically. MapViewController is created by code.

Add UISegmentedControl to MapViewController

It allows the user to choose between a discrete set of options. To allow the user switch between map types: standard, hybrid, and satellite.

Open MapViewController.swift add the following code:

```
view=mapView
let segmentedControl=UISegmentedControl(items:["Standard", "Hybrid", "Satellite"])
segmentedControl.backgroundColor=UIColor.white.withAlphaComponent(0.5)
segmentedControl.selectedSegmentIndex=0
segmentedControl.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(segmentedControl)
}
```

```
WorldTrotter > WorldTrotter > MapViewController.swift > loadView()
```

```
3 // WorldTrotter
4 //
5 // Created by daria tsoupikova on 2/3/19.
6 // Copyright © 2019 daria tsoupikova. All rights reserved.
7 //
8
9 import UIKit
10 import MapKit
11 class MapViewController: UIViewController {
12
13     var mapView: MKMapView!
14     override func loadView() {
15         //create a map view
16         mapView = MKMapView ()
17         //set it as *the* view of this view controller
18         view=mapView
19
20         let segmentedControl=UISegmentedControl(items:["Standard", "Hybrid", "Satellite"])
21         segmentedControl.backgroundColor=UIColor.white.withAlphaComponent(0.5)
22         segmentedControl.selectedSegmentIndex=0
23         segmentedControl.translatesAutoresizingMaskIntoConstraints = false
24         view.addSubview(segmentedControl)
25     }
26     override func viewDidLoad() {
27         super.viewDidLoad()
28
29         print("MapViewController loaded its view")
30     }
31 }
```

Every view has autoresizing mask

Constraints are created by default and added to the view  
Can conflict with with IB layout constraints

**`segmentedControl.translatesAutoresizingMaskIntoConstraints = false`**

The above command turns off default constraints

To use AutoLayout in code, use anchors to create constraints

**Anchors** are properties of the view that correspond to to attributes you constrain to anchor on another view

- The top anchor of segmented control should be equal to the top anchor of its superview
- The leading anchor of segmented control should be equal to the leading anchor of its superview
- The trailing anchor of the segmented control should be equal to the trailing anchor of its superview

In MapViewController.swift:

Add method **constraint(equalTo: )** to create a constraint between the two anchors.

```
view.addSubview(segmentedControl)
```

```
let topConstraint=segmentedControl.topAnchor.constraint(equalTo: view.topAnchor)
```




```
let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo:  
view.leadingAnchor)
```

```
let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo:  
view.trailingAnchor)
```

```
}
```

Add method **constraint(equalTo:)** to create a constraint between the two anchors.

```
let segmentedControl=UISegmentedControl(items:["Standard", "Hybrid", "Satellite"])
segmentedControl.backgroundColor=UIColor.white.withAlphaComponent(0.5)
segmentedControl.selectedSegmentIndex=0
segmentedControl.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(segmentedControl)
```

```
let topConstraint=segmentedControl.topAnchor.constraint(equalTo: view.topAnchor)  Initialization...
let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo:
view.leadingAnchor) 
let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo:
view.trailingAnchor) 
```

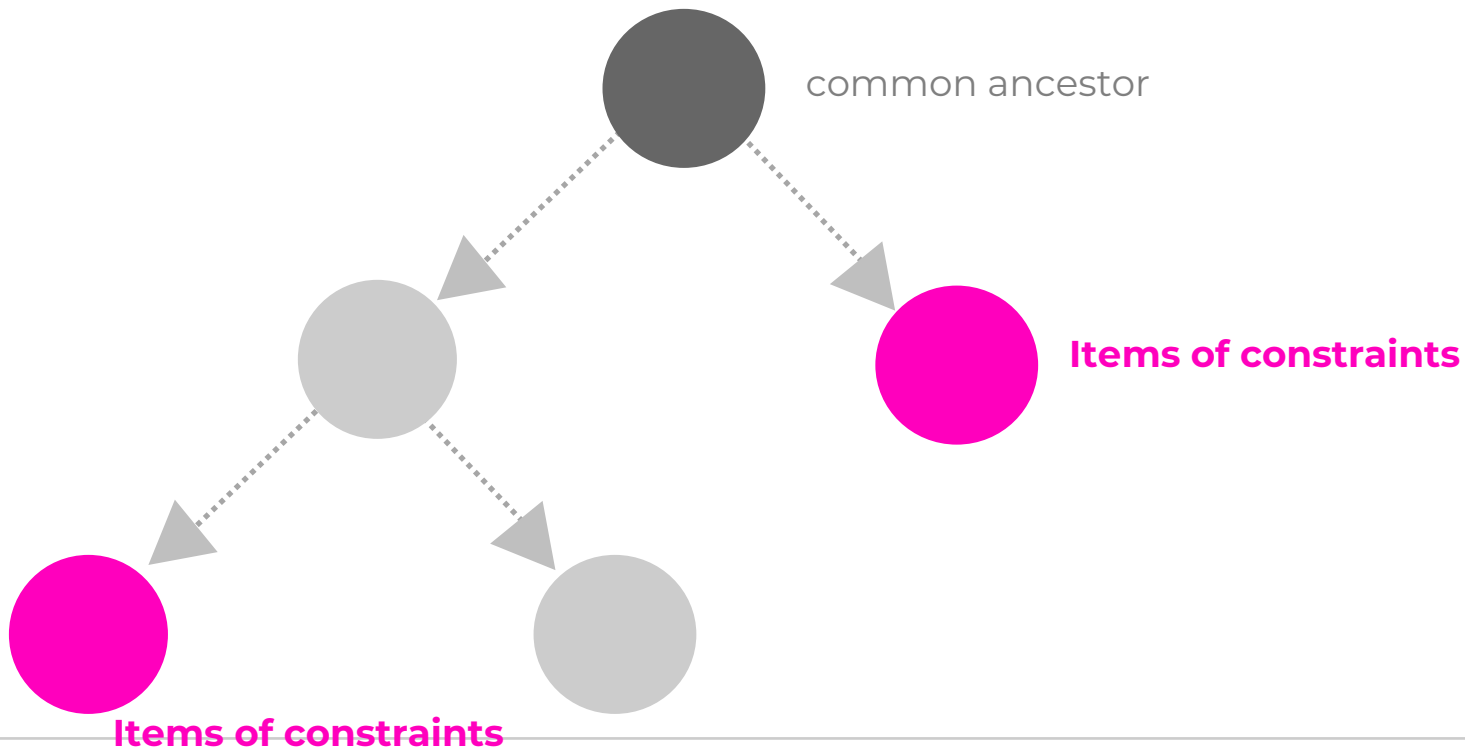
```
}
```

Because constraints are not active, the Xcode issues yellow warnings.  
To activate the constraints add:

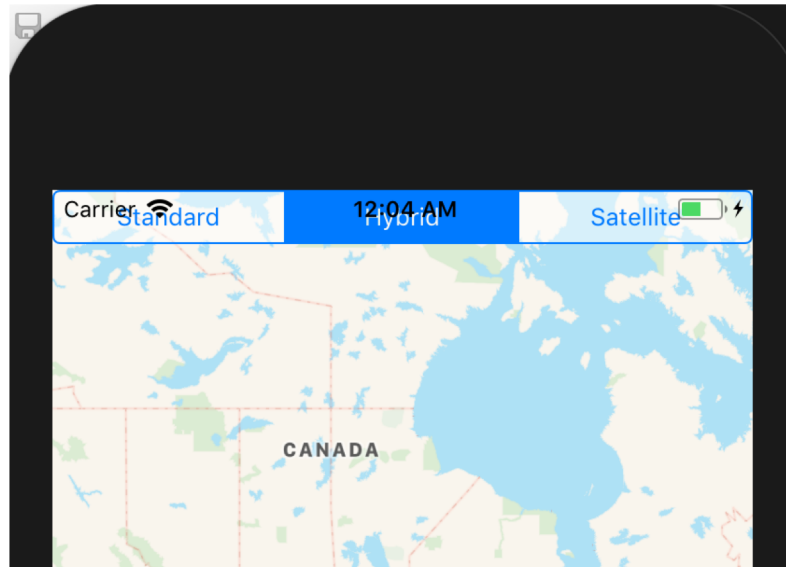
**topConstraint.isActive = true**  
**leadingConstraint.isActive = true**  
**trailingConstraint.isActive = true**

```
26         let topConstraint=segmentedControl.topAnchor.constraint(equalTo:
           view.topAnchor)
27         let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo:
           view.leadingAnchor)
28         let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo:
           view.trailingAnchor)
29
30
31         topConstraint.isActive = true
32         leadingConstraint.isActive = true
33         trailingConstraint.isActive = true
```

Constraints are added to the most recent common ancestor for the views associated with the constraint.



The segmented control is overlapping the status bar.



To assist with Layout content use two methods:

**topLayoutGuide** and **bottomLayoutGuide**

topLayoutGuide allows the content not to underlap the status bar

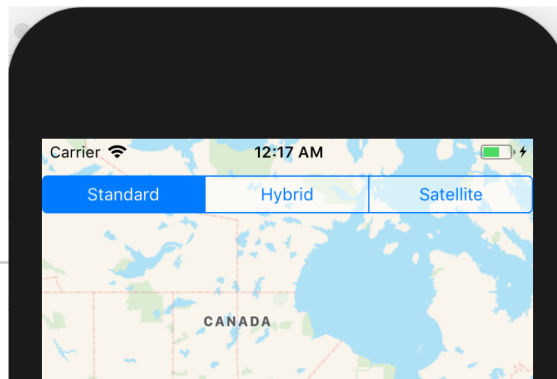
bottomLayoutGuide allows not to overlap the bottom of the screen

**let topConstraint=segmentedControl.topAnchor.constraint(equalTo:  
view.safeAreaLayoutGuide.topAnchor, constant: 8)**

```
24     view.addSubview(segmentedControl)
25
26     /*let topConstraint=segmentedControl.topAnchor.constraint(equalTo: topLayoutGuide.bottomAnchor,
27         constant: 8) was deprecated in Xcode11+ */
28
29     let topConstraint=segmentedControl.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor,
30         constant: 8)
31
32     let leadingConstraint=segmentedControl.leadingAnchor.constraint(equalTo: view.leadingAnchor)
33     let trailingConstraint=segmentedControl.trailingAnchor.constraint(equalTo: view.trailingAnchor)
34
35     topConstraint.isActive = true
36     leadingConstraint.isActive = true
37     trailingConstraint.isActive = true
38
39
```

## Build and run

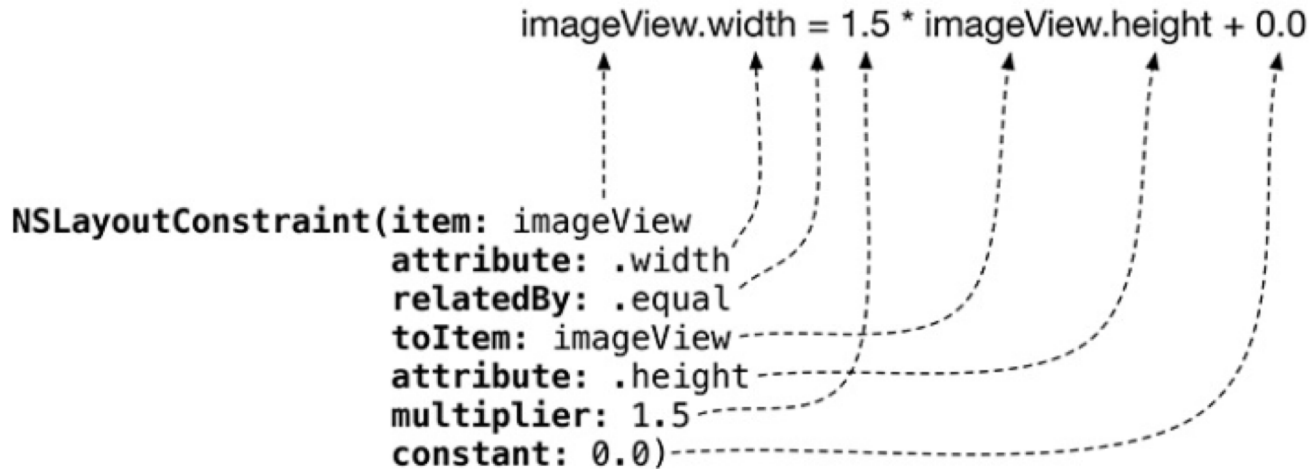
The views adapt to show status bar



The layout attributes are defined as constants in the NSLayoutConstraint class:

- NSLayoutConstraint.left
- NSLayoutConstraint.right
- NSLayoutConstraint.top
- NSLayoutConstraint.bottom
- NSLayoutConstraint.width
- NSLayoutConstraint.height
- NSLayoutConstraint.baseline
- NSLayoutConstraint.centerX
- NSLayoutConstraint.centerY
- NSLayoutConstraint.leading
- NSLayoutConstraint.trailing
- NSLayoutConstraint.lastBaseline

```
let aspectConstraint = NSLayoutConstraint (item:imageView,  
    attribute:.width,  
    relatedBy:.equal,  
    toItem:imageView,  
    attribute: .height,  
    multiplier:1.5,  
    constant:0.0);
```



Common control events:

**UIControlEvents: touchDown** – a touch down on the control

**UIControlEvents: touchUpInside** – a touch down followed by touch up within boundaries

**UIControlEvents: valueChanged** – a touch that changes the value

**UIControlEvents: editingChanged** – a touch that causes an editing change for UITextField

Change the map type when the user taps on a segment.

In `MapViewcontroller.swift` update `loadView()` to include `.valueChanged` event:

```
@objc func mapTypeChanged(_ segControl: UISegmentedControl) {  
    switch segControl.selectedSegmentIndex {  
    case 0:  
        mapView.mapType = .standard  
    case 1:  
        mapView.mapType = .hybrid  
    case 2:  
        mapView.mapType = .satellite  
    default:  
        break  
    }  
}
```

Change the map type when the user taps on a segment.

In `MapViewcontroller.swift` update `loadView()` to include `.valueChanged` event:

```
@objc func mapTypeChanged(_ segControl: UISegmentedControl) {
```

Compatibility with Objective C – newer versions of Xcode

In the textbook this code is omitted

Change the map type when the user taps on a segment.

In `MapViewcontroller.swift` update `loadView()` to include `.valueChanged` event:

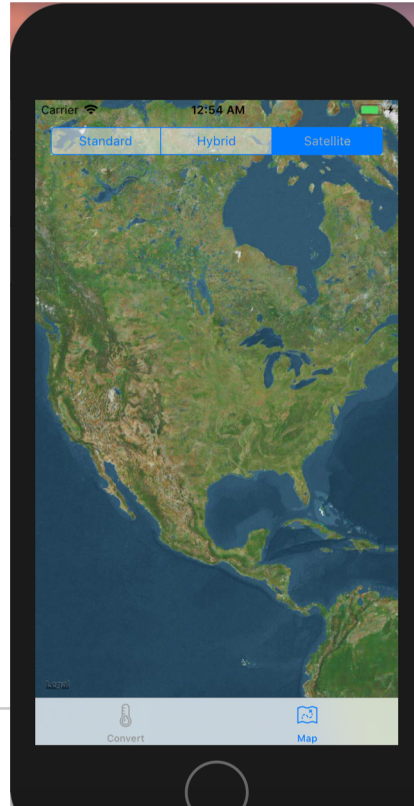
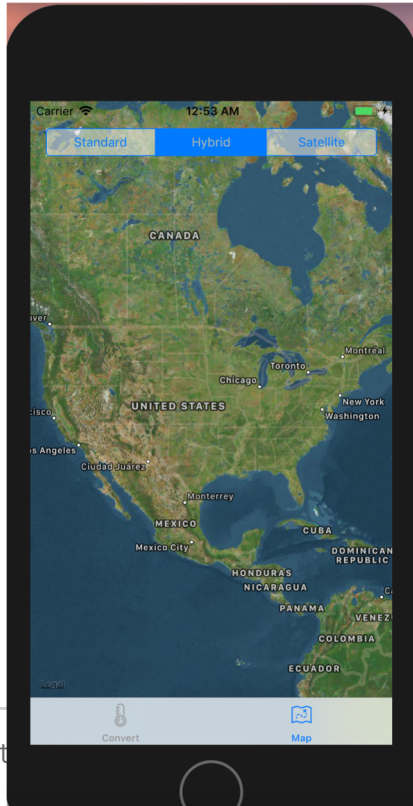
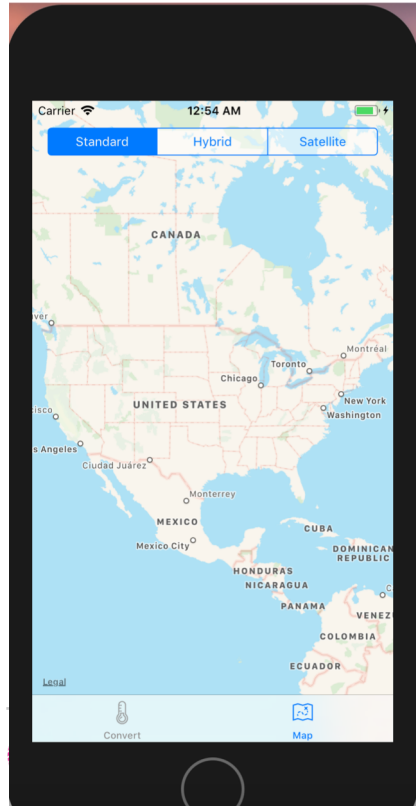
```
segmentedControl.selectedSegmentIndex = 0
```

```
segmentedControl.addTarget(self,  
    action: #selector(MapViewController.mapTypeChanged(_:)),  
    for: .valueChanged)
```

```
segmentedControl.translatesAutoresizingMaskIntoConstraints = false  
view.addSubview(segmentedControl)
```

```
22 // Set it as *the* view of this view controller
23 view = mapView
24
25 let segmentedControl = UISegmentedControl(items: ["Standard", "Hybrid", "Satellite"])
26 segmentedControl.backgroundColor = UIColor.white.withAlphaComponent(0.5)
27 segmentedControl.selectedSegmentIndex = 0
28
29 segmentedControl.addTarget(self,
30                             action: #selector(MapViewController.mapTypeChanged(_:)),
31                             for: .valueChanged)
32
33 segmentedControl.translatesAutoresizingMaskIntoConstraints = false
34 view.addSubview(segmentedControl)
35
36 let topConstraint = segmentedControl.topAnchor.constraint(equalTo: topLayoutGuide.bottomAnchor, constant: 8) ⚠️
37 let margins = view.layoutMarginsGuide
38 let leadingConstraint = segmentedControl.leadingAnchor.constraint(equalTo: margins.leadingAnchor)
39 let trailingConstraint = segmentedControl.trailingAnchor.constraint(equalTo: margins.trailingAnchor)
40
41 topConstraint.isActive = true
42 leadingConstraint.isActive = true
43 trailingConstraint.isActive = true
44 }
45
46
47 @objc func mapTypeChanged(_ segControl: UISegmentedControl) {
48     switch segControl.selectedSegmentIndex {
49     case 0:
50         mapView.mapType = .standard
51     case 1:
52         mapView.mapType = .hybrid
53     case 2:
54         mapView.mapType = .satellite
55     default:
56         break
57     }
58 }
```

## Compile, run and test



```
override func viewDidLoad() {  
    super.viewDidLoad()  
  
    let coordinate = CLLocationCoordinate2D(latitude: 41.869862, longitude: -87.648946)  
  
    let region = MKCoordinateRegion(  
        center: coordinate,  
        span: MKCoordinateSpan(latitudeDelta: 0.01, longitudeDelta: 0.01)  
    )  
  
    mapView.setRegion(region, animated: true)  
  
    print("MapViewController loaded its view.")  
}
```